

## Laboratorio Nro. 2

### Notación O grande

### Objetivos

1. Usar ecuaciones de recurrencia para determinar la complejidad en tiempo y espacio y definidos de forma recursiva
2. Usar la notación O para encontrar formalmente la complejidad asintótica en espacio y memoria de algoritmos
3. Realizar estudios empíricos para validar una hipótesis sobre el comportamiento en tiempo de ejecución de un algoritmo con varios tamaños de problema.

### Consideraciones iniciales

#### Leer la Guía



Antes de comenzar a resolver el presente laboratorio, leer la ***“Guía Metodológica para la realización y entrega de laboratorios de Estructura de Datos y Algoritmos”*** que les orientará sobre los requisitos de entrega para este y todos los laboratorios, las rúbricas de calificación, el desarrollo de procedimientos, entre otros aspectos importantes.

#### Registrar Reclamos



En caso de tener **algún comentario** sobre la nota recibida en este u otro laboratorio, pueden **enviarlo** a través de <http://bit.ly/2g4TTKf>, el cual será atendido en la menor brevedad posible.

**Traducción de Ejercicios**

En el GitHub del docente, encontrarán la traducción al español de los enunciados de los Ejercicios en Línea.

**Visualización de  
Calificaciones**

A través de **Eafit Interactiva** encontrarán **un enlace** que les permitirá **ver un registro de las calificaciones** que **emite el docente** para cada taller de laboratorio y según las rubricas expuestas. **Véase sección 3, numeral 3.7.**

**GitHub**

Crear un repositorio en su cuenta de GitHub con el nombre `st0245-eafit-suCodigoAqui`. **2.** Crear una carpeta dentro de ese repositorio con el nombre `laboratorios`. **3.** Dentro de la carpeta `laboratorios`, crear una carpeta con nombre `lab02`. **4.** Dentro de la carpeta `lab02`, crear tres carpetas: `informe`, `codigo` y `ejercicioEnLinea`. **5.** Subir el informe pdf a la carpeta `infome`, el código del ejercicio 1 a la carpeta `codigo` y el código del ejercicio en línea a la carpeta `ejercicioEnLinea`. Así:

```
st0245-eafit-suCodigoAqui
  laboratorios
    lab01
      informe
      codigo
      ejercicioEnLinea
    lab02
      ...
```

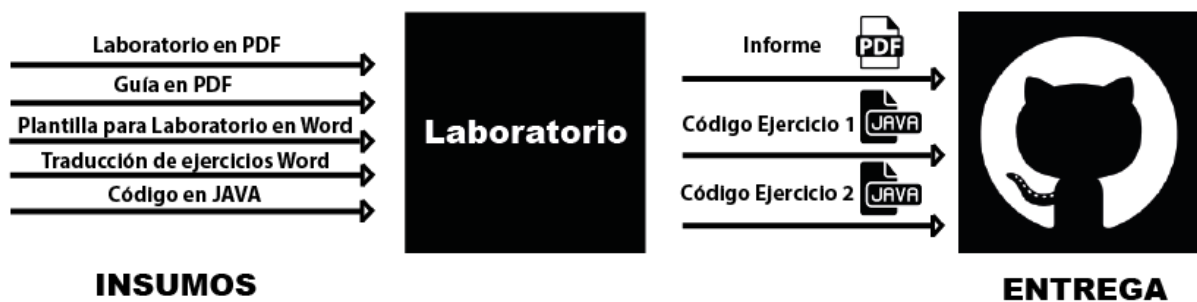
**DOCENTE MAURICIO TORO BERMÚDEZ**

Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627

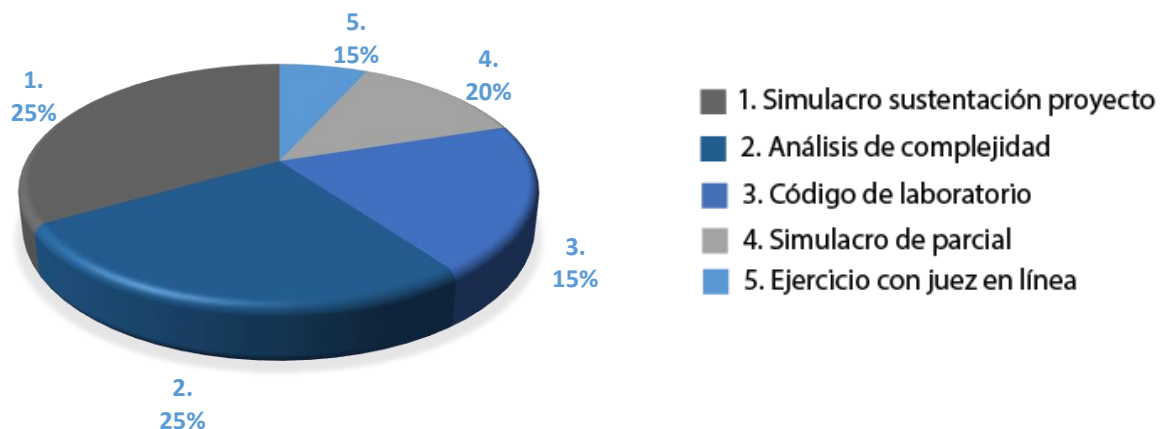
Correo: [mtorobe@eafit.edu.co](mailto:mtorobe@eafit.edu.co)

## Intercambio de archivos

Los archivos que **ustedes deben entregar** al docente son: **un archivo PDF** con el informe de laboratorio usando la plantilla definida, y **dos códigos**, uno con la solución al numeral 1 y otro al numeral 2 del presente. Todo lo anterior se entrega en **GitHub**.



## Porcentajes y criterios de evaluación para el laboratorio



## Ejercicios a resolver

### 1. Códigos para entregar en GitHub:



En la vida real, la documentación de software hace parte de muchos estándares de calidad como CMMI e ISO/IEC 9126



Véase Guía **en Sección 3, numeral 3.4**



Código de laboratorio en **GitHub**. Véase Guía en **Sección 4, numeral 4.24**



*Es opcional entregar documentación. Si lo hace, utilice **javadoc** o equivalente. No suba el HTML a GitHub.*









**No se reciben** archivos en **.RAR** ni en **.ZIP**

#### Para el ejercicio 1.1:

- Identifiquen qué representa el tamaño del problema para cada algoritmo
- Identifiquen valores apropiados para el tamaño del problema
- Tomen los tiempos para 20 tamaños del problema diferentes.
- Hagan una gráfica en Excel para cada algoritmo y pasarla a Word luego
- Entreguen el informe en formato PDF

**1.1 Extiendan el código existente para medir los tiempos de *Insertion Sort* y *Merge sort* con 20 tamaños del problema.** Si para todos los tamaños del problema se demora 0 ms, no son adecuados.

## 2) Ejercicios en línea sin documentación HTML en GitHub

	Véase Guía en <b>Sección 3, numeral 3.3</b>		<b>No entregar documentación HTML</b>
	Entregar un archivo en <b>.JAVA</b>		<b>No se reciben archivos en .PDF</b>
	<b>Resolver</b> los problemas de <b>CodingBat</b> usando <b>Recursión</b>		Código del ejercicio en línea en <b>GitHub</b> . Véase Guía en <b>Sección 4, numeral 4.24</b>



**NOTA 1:** Recuerden que, *si toman la respuesta de alguna fuente, deben referenciar según el tipo de cita correspondiente*. Véase Guía en **Sección 4, numerales 4.16 y 4.17**

**2.1** Resuelvan mínimo 5 ejercicios del nivel **Array 2** de la página **CodingBat**:  
<http://codingbat.com/java/Array-2>



**NOTA 1:** La complejidad máxima para ejercicios de **Array2** es  $O(n)$

**2.2** Resuelvan mínimo 5 ejercicios del nivel **Array 3** de la página **CodingBat**:  
<http://codingbat.com/java/Array-3>



**NOTA:** La complejidad máxima para ejercicios de **Array 3** es  $O(n^2)$

## 3) Simulacro de preguntas de sustentación de Proyectos

DOCENTE MAURICIO TORO BERMÚDEZ  
Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627  
Correo: [mtorobe@eafit.edu.co](mailto:mtorobe@eafit.edu.co)



Véase Guía en **Sección 3,**  
**Numeral 3.5**



Entregar informe de  
laboratorio en **PDF**



Usen la **plantilla** para  
responder laboratorios



**No apliquen Normas**  
**Icontec** para esto

**3.1** De acuerdo a lo realizado en el numeral 1.1, construya una tabla donde muestre, para cada algoritmo, cuánto se demora para cada uno de los 20 valores del tamaño del problema.



En la vida real, las gráficas son una forma simple y concisa de representar resultados cuantitativos en Ingeniería de Sistemas

**3.2** Grafiquen los tiempos que tomó en cada algoritmo para los diferentes tamaños del problema. Grafiquen el Tamaño de la Entrada Vs. Tiempo de Ejecución.



En la vida real, bases de datos relacionales como **MySQL** guardan los datos ordenados, usando como criterio la llave primaria. Como las bases de datos pueden almacenar millones de elementos, es importante entender la eficiencia de los algoritmos de ordenamiento.

**3.3** Teniendo en cuenta lo anterior, ¿Qué tan eficiente es *merge sort* con respecto a *insertion sort* para arreglos grandes? ¿Es apropiado usar *insertion sort* para una base de datos con millones de elementos?

**3.4** Expliquen con sus propias palabras cómo funciona el ejercicio de Array 3 de Coding Bat llamado *maxSpan* y ¿por qué?

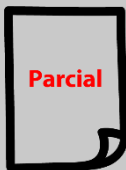


**NOTA 1:** Recuerden que debe explicar su implementación en el informe PDF

**3.5** Calculen la complejidad de los ejercicios en línea, numerales 2.1 y 2.2, y agréguelas al informe PDF

**3.6** Expliquen con sus palabras las variables (qué es 'n', qué es 'm', etc.) del cálculo de complejidad del numeral anterior

#### 4) Simulacro de Parcial en el informe PDF



Para este simulacro, agreguen ***sus respuestas*** en el informe PDF.

Resuelva, como mínimo, los ejercicios marcados con **color rojo**.



***El día del Parcial no tendrán computador, JAVA o acceso a internet.***

**1.** Supongamos que  $P(n,m)$  es una función cuya complejidad asintótica es  $O(n \times m)$  y  $H(n,m)$  es otra función cuya complejidad asintótica es  $O(m \times A(n,m))$ , donde  $A(n,m)$  es otra función.

¿Cuál de las siguientes funciones, definitivamente, **NO** podría ser la complejidad asintótica de  $A(n,m)$  si tenemos en cuenta que  $P(n,m) > H(n,m)$  para todo  $n$  y para todo  $m$ ?

**a)**  $O(\log n)$

- b)  $O(\sqrt{n})$
- c)  $O(n+m)$
- d)  $O(1)$

2. Dayla sabe que la complejidad asintótica de la función  $P(n)$  es  $O(\sqrt{n})$ . Ayúdenle a Dayla a sacar la complejidad asintótica para la función  $\text{mystery}(n,m)$ .

```
void mystery(int n, int m) {  
    for(int i = 0; i < m; ++i){  
        boolean can = P(n);  
        if(can)  
            for(int i = 1; i * i <= n; i++)  
                //Hacer algo en  $O(1)$   
        else  
            for(int i = 1; i <= n; i++)  
                //Hacer algo en  $O(1)$   
    }  
}
```

La complejidad de  $\text{mystery}(n,m)$  es:

- a)  $O(m+n)$
- b)  $O(m \times n \times \sqrt{n})$
- c)  $O(m+n+\sqrt{n})$
- d)  $O(m \times n)$

3. El siguiente algoritmo imprime todos los valores de una matriz. El tamaño de la matriz está definida por los parámetros largo y ancho. Teniendo en cuenta que el largo puede ser como máximo 30, mientras que el ancho puede tomar cualquier valor, ¿Cuál es su complejidad asintótica en el peor de los casos del algoritmo?

```
01 public void matrices(int[][] m, int largo, int ancho)  
02     for(int i=0; i < largo; i++)  
03         for(int j=0; j < ancho; j++)  
04             print (m[i][j]);
```

- a)  $O(\text{largo})$



- b)  $O(\text{ancho})$
- c)  $O(\text{largo} + \text{ancho})$
- d)  $O(\text{ancho} \times \text{ancho})$
- e)  $O(1)$

4. Sabemos que  $P(n)$  ejecuta  $n^3 + n$  pasos y que  $D(n)$  ejecuta  $n+7$  pasos. ¿Cuál es la complejidad asintótica, en el peor de los casos, de la función  $B(n)$ ?

```
public int B(int n)
    int getP = P(n);
    int ni = 0;
    for(int i = 0; i < n; ++i)
        if(D(i) > 100){
            ni++;
        }
    int nj = getP + D(n) * ni;
    return nj;
```

- a)  $O(n^4)$
- b)  $O(n^3)$
- c)  $O(n^2)$
- d)  $O(2^n)$

5. El siguiente algoritmo calcula las tablas de multiplicar del 1 a  $n$ . ¿Cuál es su complejidad asintótica en el peor de los casos?

```
public void tablas(int n)
    for(int i=0; i < n; i++)
        for(int j=0; j < n; j++)
            print (i+"*"+j+"="+i*j);
```

- a)  $O(n)$
- b)  $O(2^i)$
- c)  $O(i)$
- d)  $O(n^2)$

6. Consideren el siguiente algoritmo:

```
1 public int misterio(int n, int i){
2     if (i >= n){
3         return n;
4     }
5     return n * misterio(n, i + 1);
6 }
```

¿Cuántas instrucciones ejecuta el algoritmo en el peor de los casos?

- a)  $T(n)=T(n-1)+C$
- b)  $T(n)=T(n-1)+T(n-2)+C$
- c)  $T(n)=T(n/2)+C$
- d)  $T(n)=T(n+1)+C$

7. El siguiente algoritmo cuenta del  $n$  al 1.

```
public void imprimir(int n) {
    if (n == 1) println(1);
    else { println(n);
          imprimir(n-1); }}
```

7.1 ¿Cuál es el número de pasos que ejecuta para el peor de los casos?

$T(n)=$  \_\_\_\_\_

7.2 ¿Cuál es la complejidad asintótica en el peor de los casos?

$O(\text{_____})$

8. Asuma que  $func(n)$  ejecuta exactamente  $T(n)=\sqrt{n}$  pasos. ¿Cuál de las siguientes afirmaciones es correcta?

```
void misterio(int n){
```

```
for(int i=1;i+i<=n;i=i+1){  
    int k = func(i);  
}  
}
```

- (a) La función *misterio(n)* ejecuta  $O(n^3)$  pasos.
- (b) La función *misterio(n)* ejecuta  $O(n\sqrt{n})$  pasos.
- (c) La función *misterio(n)* ejecuta  $O(n^2)$  pasos.
- (d) La función *misterio(n)* ejecuta  $O(n)$  pasos.

9. ¿Cuál de las siguientes afirmaciones es correcta con respecto a la función *func1(n,m)*?

```
void func1(int n, int m){  
    for(int i = 0; i < n; i++){  
        for(int j = 0; j < n + m; j++){  
            print(j);  
        }  
    }  
}
```

- (a) Ejecuta menos de  $n+m$  pasos.
- (b) Ejecuta exactamente  $n+m$  pasos.
- (c) Ejecuta  $n \times m$  pasos.
- (d) Ejecuta más de  $n \times m$  pasos.

10. ¿Cuál de las siguientes afirmaciones es correcta con respecto a la función *func2(n)*?

```
void func2(int n){  
    for(int i = 2; i * i <= n; i++){  
        for(int k = 2; k * k <= n; k++){  
            print(j);  
        }  
    }  
}
```

- (a) Ejecuta más de  $n^2$  pasos.
- (b) Ejecuta más de  $n^3$  pasos.

(c) Ejecuta menos de  $n \cdot \log n$  pasos.

(d) Ejecuta exactamente  $n^2$  pasos.

**11.** ¿Cuál de las siguientes afirmaciones es correcta con respecto a  $\text{func3}(n)$ ?

```
int func3(int n){  
    if(n == 1 || n == 2){  
        return n;  
    }  
    int ni = func3(n - 1);  
    int nj = func3(n - 2);  
    int suma = ni + nj;  
    return suma;  
}
```

(a) Ejecuta  $T(n) = T(n-1) + c$  pasos.

(b) Ejecuta  $T(n) = T(n-1) + cn$  pasos.

(c) Ejecuta  $T(n) = T(n-1) + T(n-2) + c$  pasos.

(d) Ejecuta  $T(n) = T(n/2) + c$  pasos.

**12.** Sea  $f(n, m) = O(m \cdot \log n)$  y  $g(n, m) = O(m \sqrt{n})$ . Siempre se cumple que  $m \geq n$ . ¿Cuál es la complejidad asintótica de  $h(n, m) = O(f(n, m) + g(n, m))$ ?

(a)  $O(m^2 \log n \sqrt{n})$

(b)  $O(m \sqrt{n})$

(c)  $O(m + n)$

(d)  $O(n + m + \log n \sqrt{n})$

**13.** Sea  $f(n, m) = O(n + m)$  y  $g(n, m) = O(n^2 + m)$ . Siempre se cumple que  $n \geq m$ . ¿Cuál es la solución de  $h(n, m) = O(f(n, m) \times g(n, m))$ ?

(a)  $O(n^3)$

(b)  $O(n^2 + m)$

(c)  $O(n + m)$

(d)  $O(nm + m^2)$

## 5. [Ejercicio Opcional] Lectura recomendada



"Quienes se preparan para el ejercicio de una profesión requieren la adquisición de competencias que necesariamente se sustentan en procesos comunicativos. Así cuando se entrevista a un ingeniero recién egresado para un empleo, una buena parte de sus posibilidades radica en su capacidad de comunicación; pero se ha observado que esta es una de sus principales debilidades..."

Tomado de <http://bit.ly/2gJKzJD>



Véase Guía en **Sección 3, numeral 3.5 y 4.20** de la Guía Metodológica, "Lectura recomendada" y "Ejemplo para realización de actividades de las Lecturas Recomendadas", respectivamente

Posterior a la lectura del texto "**R.C.T Lee et al., Introducción al análisis y diseño de algoritmos, Capítulo 2. 2005**" realicen las siguientes actividades que les permitirán sumar puntos adicionales:

- a) Escriban un resumen de la lectura que tenga una longitud de 100 a 150 palabras
- b) Hagan un mapa conceptual que destaque los principales elementos teóricos.



**NOTA 1:** Si desean una lectura adicional en idioma español, consideren la siguiente: "**John Hopcroft et al., Fundamentos de Algoritmia. Capítulo 3: Notación Asintótica. Páginas 11– 98. 1983**" que pueden encontrarla en biblioteca.



**NOTA 2:** Estas respuestas también deben incluirlas en el informe PDF

## 6. [Ejercicio Opcional] Trabajo en Equipo y Progreso Gradual



El trabajo en equipo es una exigencia actual del mercado. "Mientras algunos medios retratan la programación como un trabajo solitario, la realidad es que requiere de mucha comunicación y trabajo con otros. Si trabajas para una compañía, serás parte de un equipo de desarrollo y esperarán que te comuniques y trabajes bien con otras personas"

Tomado de <http://bit.ly/1B6hUDp>



Véase Guía en **Sección 3, numeral 3.6** y **Sección 4, numerales 4.21 y 4.22 y 4.23** de la Guía Metodológica

- a) Entreguen copia de todas las actas de reunión usando el tablero Kanban, con fecha, hora e integrantes que participaron
- b) Entreguen el reporte de *git*, *svn* o *mercurial* con los cambios en el código y quién hizo cada cambio, con fecha, hora e integrantes que participaron
- c) Entreguen el reporte de cambios del informe de laboratorio que se genera *Google docs* o herramientas similares



**NOTA:** Estas respuestas también deben incluirlas en el informe PDF

## 7. [Ejercicio Opcional] Laboratorio en inglés:



El inglés es un idioma muy importante en la Ingeniería de Sistemas porque la mayoría de los avances en tecnología se publican en este idioma y la traducción, usualmente se demora un tiempo y es sólo un resumen de la información original.

Adicionalmente, dominar el inglés permite conseguir trabajos en el exterior que son muy bien remunerados

Tomado de [goo.gl/4s3LmZ](https://goo.gl/4s3LmZ)

Entreguen el código y el informe traducido al inglés. Utilicen la plantilla dispuesta en este idioma para el laboratorio

## Resumen de ejercicios a resolver


**1.1** Extiendan el código existente para medir los tiempos de *Insertion Sort* y *Merge sort*

**2.1** Resuelvan mínimo 5 ejercicios del nivel *Array 2* de la página *CodingBat*:  
<http://codingbat.com/java/Array-2>

**2.2** Resuelvan mínimo 5 ejercicios del nivel *Array 3* de la página *CodingBat*:  
<http://codingbat.com/java/Array-3>

**3.1** De acuerdo a lo realizado en el numeral 1.1, construya una tabla donde muestre, para cada algoritmo, cuánto se demora para cada uno de los tamaños del problema.

**3.2** Grafiquen los tiempos que tomó en cada algoritmo para los diferentes tamaños del problema.

	UNIVERSIDAD EAFIT ESCUELA DE INGENIERÍA DEPARTAMENTO DE INFORMÁTICA Y SISTEMAS	Cód. ST0245
		Estructuras de Datos 1

**3.3** ¿Qué tan eficiente es *merge sort* con respecto a *insertion sort* para arreglos grandes? ¿Es apropiado usar *insertion sort* para una base de datos con millones de elementos? ¿Por qué?

**3.4** Expliquen con sus propias palabras cómo funciona su solución del ejercicio de Array 3 de Coding Bat llamado *maxSpan*

**3.5** Calculen la complejidad de los ejercicios en línea, numerales 2.1 y 2.2, y agréguela al informe PDF

**3.6** Expliquen con sus palabras las variables (qué es 'n', qué es 'm', etc.) del cálculo de complejidad del numeral anterior

**4.** Simulacro de Parcial

**5.** Lectura recomendada **[Ejercicio Opcional]**

**6.** Trabajo en Equipo y Progreso Gradual **[Ejercicio Opcional]**

**7.** Entreguen el código y el informe en inglés. Utilicen la plantilla dispuesta en este idioma para el laboratorio. **[Ejercicio Opcional]**



## Ayudas para resolver los ejercicios

Ayudas para el Ejercicio 1.....	<a href="#"><u>Pág. 19</u></a>
Ayudas para el Ejercicio 1.1.....	<a href="#"><u>Pág. 19</u></a>
Ayudas para el Ejercicio 3.5.....	<a href="#"><u>Pág. 19</u></a>
Ayudas para el Ejercicio 3.11.....	<a href="#"><u>Pág. 20</u></a>
Ayudas para el Ejercicio 4.....	<a href="#"><u>Pág. 20</u></a>
Ayudas para el Ejercicio 5a.....	<a href="#"><u>Pág. 20</u></a>
Ayudas para el Ejercicio 5b.....	<a href="#"><u>Pág. 21</u></a>
Ayudas para el Ejercicio 6a.....	<a href="#"><u>Pág. 21</u></a>
Ayudas para el Ejercicio 6b.....	<a href="#"><u>Pág. 21</u></a>
Ayudas para el Ejercicio 6c.....	<a href="#"><u>Pág. 21</u></a>

## Ayudas para el Ejercicio 1



**PISTA 1:** Lean las diapositivas tituladas “**Data Structures I: O Notation**”, donde encontrarán algoritmos, y en **Eafit Interactiva**, las implementaciones en Java de cada uno



**PISTA 2:** Si decide hacer la documentación, consulten la *Guía en Sección 4, numeral 4.1 “Cómo escribir la documentación HTML de un código usando JavaDoc”*

## Ayudas para el Ejercicio 1.1



**PISTA 1:** Véase **Guía sección 4, numeral 4.6 “Cómo usar la escala logarítmica en Microsoft Excel”**.



**PISTA 3:** Véase **Guía sección 4, numeral 4.4 “Cómo aumentar el tamaño del heap y del stack en Java”**



**PISTA 4:** Véase **Guía sección 4, numeral 4.5 “Cómo visualizar el montículo (heap) y el stack, y el consumo total de memoria de Java”**

## Ayudas para el Ejercicio 3.5



**PISTA 1:** Véase **Guía sección 4, numeral 4.6 “Cómo usar la escala logarítmica en Microsoft Excel”**.



**PISTA 2:** Si todos los tiempos de un algoritmo dan más de 5 minutos, realice otra tabla, para ese algoritmo, tomando tiempos para arreglos de tamaño 1000, 10000 y 100000.



**PISTA 3:** Véase **Guía sección 4, numeral 4.4 “Cómo aumentar el tamaño del heap y del stack en Java”**



**PISTA 4:** Véase **Guía sección 4, numeral 4.5** “Cómo visualizar el montículo (heap) y el stack, y el consumo total de memoria de Java”

## Ayudas para el Ejercicio 3.11



**PISTA 1:** Véase **Guía en Sección 4, numeral 4.11** “Cómo escribir la complejidad de un ejercicio en línea”



**PISTA 2:** Véase **Guía en Sección 4, numeral 4.19** “Ejemplos para calcular la complejidad de un ejercicio de CodingBat”

## Ayudas para el Ejercicio 4



**PISTA 1:** Véase **Guía en Sección 4, Numeral 4.18** “Respuestas del Quiz”



**PISTA 2:** Lean las diapositivas tituladas “**Data Structures I: O notation**” y encontrarán la mayoría de las respuestas”

## Ayudas para el Ejercicio 5a



**PISTA 1:** En el siguiente enlace, unos consejos de cómo hacer un buen resumen <http://bit.ly/2knU3Pv>



**PISTA 2:** [Aquí](#) le explican cómo contar el número de palabras en Microsoft Word

## Ayudas para el Ejercicio 5b



**PISTA 1:** Para que hagan el mapa conceptual se recomiendan herramientas como las que encuentran en <https://cacoo.com/> o <https://www.mindmup.com/#m:new-a-1437527273469>

## Ayudas para el Ejercicio 6a



**PISTA 1:** Véase **Guía en Sección 4, Numeral 4.21** “Ejemplo de cómo hacer actas de trabajo en equipo usando Tablero Kanban”

## Ayudas para el Ejercicio 6b



**PISTA 1:** Véase **Guía en Sección 4, Numeral 4.23** “Cómo generar el historial de cambios en el código de un repositorio que está en svn”

## Ayudas para el Ejercicio 6c



**PISTA 1:** Véase **Guía en Sección 4, Numeral 4.22** “Cómo ver el historial de revisión de un archivo en Google Docs”